Proyecto de Innovación y Mejora de la Calidad Docente
Convocatoria 2015

Num. de proyecto: 290

Título: Arquitecturas dinámica de redes inalámbricas en banda libre para la ejemplificación de conceptos de transmisión en aplicaciones de “Internet Of Things”

Nombre del responsable del proyecto: José Luis Ayala Rodrigo
Centro: Facultad de Informática
Departamento: Arquitectura de Computadores y Automática

1. Objetivos propuestos en la presentación del proyecto

Este proyecto de innovación educativa propone una metodología práctica y un equipamiento novedoso para la docencia de la asignatura de Redes y Servicios de Telecomunicación II impartida en tercer curso del Grado en Ingeniería Electrónica de Comunicaciones. Mediante la incorporación de elementos prácticos a la docencia como los recogidos en este proyecto, se pretende ahondar en los conceptos de uso espectral, acceso a un canal compartido, enrutamiento, topología de red, relación consumo vs. potencia de transmisión, etc. desde una perspectiva práctica que facilite el aprendizaje y despierte la curiosidad del alumnado. Para ello, se propondrá un despliegue de nodos inalámbricos, y un entorno de programación de éstos, que permita la evaluación de los contenidos antes descritos.

En particular, los objetivos perseguidos con la realización de este proyecto son los siguientes:

- Implementar una red real, no simulada, con nodos inalámbricos para aplicar sobre ella los conceptos de enrutamiento, topologías, servicios (calidad), de la asignatura Redes II y otros conocimientos transversales de asignaturas como Electrónica y Comunicaciones Inalámbricas.
- Diseño de una guía práctica para una futura implementación en la asignatura. Redacción de un manual de puesta en marcha para ayudar al alumno a crear su propio entorno.
- Puesta en marcha de una red con un número suficiente de motas inalámbricas comerciales y el entorno de programación a bajo nivel.
- Aprendizaje de uso y manipulación de los protocolos de rutado de mensajes entre los dispositivos inalámbricos comerciales.
- Despliegue de las motas para implementar distintas topologías de red: anillo, estrella, árbol, etc.
- Probar los conceptos de acceso al medio generando ocupaciones de canal para provocar una decisión de cambio del mismo.
- Probar los conceptos de potencia de transmisión (en banda libre) dando lugar a saturaciones de potencia del canal.
- Integración de un componente de escucha inalámbrica en el sistema (baliza) para visualización del espectro radioeléctrico y observar las bandas de frecuencia de emisión, los canales y las potencias de transmisión en un entorno controlado.
- Probar los conceptos de calidad de servicio generando pérdidas en una comunicación.
- Implementación de algún servicio básico (retransmisión de hora o lectura de un dato de un sensor, ya sea analógico y digital, por ejemplo) sobre el que probar la calidad de servicio.

2. Objetivos alcanzados una vez finalizado el proyecto

En general, podemos considerar que hemos alcanzado todos los objetivos marcados, salvo la aplicación del material didáctico desarrollado y su impacto en el aprendizaje de los alumnos, ya que deberá probarse en el curso académico siguiente. De la misma manera, la publicación de carácter educativo no se ha conseguido todavía, al carecer de datos reales sobre la satisfacción del alumnado con la plataforma. Sin embargo, todos los objetivos de carácter técnico sí que han sido alcanzados. A continuación enumeramos los hitos del proyecto:

- Puesta en marcha de una plataforma de red inalámbrica mediante motas Nimbus, establecimiento del entorno de desarrollo, depuración y programación.
- Puesta en marcha de un sistema de monitorización de la comunicación inalámbrica mediante SDR de la plataforma HackRF. Configuración del dispositivo como analizador de espectros y osciloscopio.
- Implementación de un demodulador GFSK coherente en GNURadio
- Integración con Smart-RF Studio para la generación de paquetes de datos
- Generación de una topología en estrella mediante la programación de 5 motas Nimbus
- Generación de una topología en anillo asíncrono mediante la programación de 5 motas Nimbus
- Generación de una topología en anillo síncrono mediante la programación de 5 motas Nimbus
- Desarrollo de 3 prácticas para los alumnos, de forma que se evalúen los conocimientos adquiridos durante la experimentación con los conceptos prácticos descritos.

3. Metodología empleada en el proyecto

- Estudio del mercado para evaluar diversas plataformas hardware que pudieran satisfacer nuestros requisitos técnicos: rangos de frecuencia, integración con el software, etc. y estuvieran en consonancia con el presupuesto solicitado/recibido.
- Estudio del software libre disponible para trabajar con diversas plataformas hardware.
- Compra del material seleccionado.
- Instalación de software de configuración y manejo del SDR HackRF y de las motas inalámbricas Nimbus
- Instalación de GNU Radio en entorno Linux y comunicación con el resto de las herramientas
- Definición y desarrollo de un entorno de diseño y programación de las motas inalámbricas basado en Eclipse
- Programación de las librerías de comunicación

- Desarrollo e implementación de práctica introductoria al entorno de trabajo
- Desarrollo e implementación de práctica experimental sobre modulaciones, nivel de potencia de transmisión y calidad de la señal recibida
- Desarrollo e implementación de práctica experimental sobre topologías de red inalámbricas
- Escritura y depuración de enunciados de prácticas para alumnos

4. Recursos humanos

Alberto A. Del Barrio es Doctor en Ingeniería Informática. Su tesis está relacionada con la síntesis de benchmarks de procesado de la señal. Es revisor frecuente de la revista Digital Signal Processing, situada en el primer cuartil y con un JCR=1.495. Actualmente es profesor y coordinador de la asignatura de Máster en Informática denominada Sistemas Empotrados Distribuidos. Además, ha impartido las siguientes asignaturas relacionadas: Laboratorio de Redes (Grado en Ingeniería Informática, Grado en Ingeniería del Software), y tiene publicaciones docentes y evaluaciones positivas por el sistema Docentia. Durante el curso 2013/2014 fue el responsable del PIMCD titulado "Receptor Software de bajo coste e Interfaz Computerizada para el Estudio Práctico de las Comunicaciones Radioeléctricas".

José Luis Ayala es Doctor Ingeniero de Telecomunicación, con doble especialidad en el área de Microelectrónica y de Procesado Digital de la Señal. Es el profesor coordinador y responsable de la implantación de la asignatura de “Teoría de la Comunicación” en la UCM, así como el coordinador de segundo curso del Grado de Ingeniería de Sistemas Electrónicos de Telecomunicación. Adicionalmente, ha impartido docencia del área de comunicaciones inalámbricas en la Univ. San Pablo CEU (“Teoría de la Comunicación”) y la Universidad Europea de Madrid (Sistemas de Telecomunicación), así como docencia de primer, segundo y tercer ciclo de la titulación de Ingeniero de Telecomunicación en la UPM, y de la titulación de Ingeniero en Informática de la UCM.

Josué Pagán es PDI joven, trabajando en su tesis doctoral en Ingeniería de Telecomunicación, donde el uso de redes inalámbricas de sensores encuentran su aplicación en el modelo robusto y flexible. En particular, Josué centra su labor investigadora en la aplicación práctica de topologías de red inalámbricas para la adquisición de datos biométricos de forma ambulatoria en pacientes, y la adquisición indoor de variables determinantes del consumo energético en centros de procesamiento de datos. Cuenta con gran experiencia en la programación de dichos entornos, y en la resolución de las cuestiones prácticas y metodológicas que un entorno como el descrito puede presentar.

Marina Zapater es PDI joven y su tesis en Ingeniería de Telecomunicación está relacionada con el modelado y la optimización de consumo a varios niveles de abstracción, desde procesadores para sistemas empotrados, servidores y centros de proceso de datos. Actualmente es profesora y coordinadora de la asignatura Comunicaciones Inalámbricas en el Grado de Electrónica de Comunicaciones de la UCM. Además, ha colaborado en la impartición del laboratorio de Redes del mismo grado, en el laboratorio de Fundamentos de

Computadores y de Tecnología y Organización de Computadores. También ha colaborado en los laboratorios de Electrónica Digital, y Circuitos Electrónicos del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la UPM.

Román Hermida Correa es Doctor en Ciencias Físicas y cuenta una experiencia docente de más de 35 años en la UCM, contando con seis quinquenios de evaluación docente favorable. A lo largo de su trayectoria ha participado activamente en el diseño de Planes de Estudios, y de manera muy particular en las dos últimas renovaciones de los de la Facultad de Informática, como Subdirector de Estudios (para los planes implantados en 1998) y como Decano (para los planes implantados en 2010). Además ha participado de forma destacada en el diseño de los laboratorios de la nueva Facultad de Informática, destacando el Laboratorio de Estructura de Computadores, donde participó no sólo en el diseño de los contenidos docentes, sino también en el diseño y fabricación de la electrónica necesaria. Igualmente, se ha responsabilizado de la implantación de los primeros laboratorios de diseño basado en FPGAs y de circuitos integrados. Esta doble visión como técnico y como gestor educativo se considera indispensable para el proyecto.

5. Desarrollo de las actividades

Las tareas del proyecto se han desarrollado con cierto retraso con respecto al cronograma original, aunque tal y como se ha comentado anteriormente todos los objetivos técnicos se han cubierto satisfactoriamente. Dicho retraso se debe fundamentalmente a la dificultad para financiar el conjunto de motas inalámbricas requeridas para la implementación de la red de comunicación y la experimentación con ésta (debido a que no se consiguió el 100% del presupuesto solicitado, hubo que utilizar una fuente de financiación adicional).

Pese a esta situación, hemos cumplido con los objetivos básicos marcados inicialmente, dejando para el futuro próximo su aplicación a la asignatura “Redes y Servicios de Telecomunicación II”, y alguna publicación docente que dé a conocer el desarrollo de la plataforma planteada y los resultados académicos obtenidos.

A continuación, vamos a listar los recursos materiales en los que se ha basado este proyecto, que son los siguientes:

- HackRF One Software Defined Radio
- Una antena telescópica.
- Adaptadores macho/hembra para la antena.
- Motas inalámbricas 868MHz
- Placas de desarrollo Nimbus
- Componentes electrónicos de apoyo.

Además, se han utilizado los ordenadores personales de todos los componentes del proyecto. Todos los ordenadores utilizados han contado con una instalación de Ubuntu y de GNURadio, adecuadamente integrada con el entorno de desarrollo Eclipse y las librerías de comunicación inalámbrica de Nimbus

Por último y para concluir, enumeraremos los resultados obtenidos en este proyecto, que son los siguientes:

- Una plataforma hardware procesadora y con capacidad de comunicación inalámbrica (mota)
- Una interfaz software integrada con la plataforma hardware, y desarrollada sobre una distribución Linux con amplio soporte, como es Ubuntu.
- Manuales de instalación del entorno y comunicación con las unidades hardware
- Material didáctico para realizar 3 prácticas en la asignatura “Redes y Servicios de Telecomunicación II”:
    - Una práctica básica, explicando los componentes básicos del entorno.
    - Una práctica sobre modulación de señal inalámbrica, potencia de transmisión e integridad de señal recibida
    - Una práctica sobre topologías de redes inalámbricas

Práctica 0: HackRF, gnuradio y motas inalámbricas

En esta práctica aprenderemos a instalar y a utilizar el entorno que emplearemos en el resto de la asignatura. En concreto, describiremos los detalles asociados a:

- Los sistemas Software Defined Radio (SDR).
- La plataforma hardware HackRF.
- El software GNU Radio para dar soporte al HackRF.
- Las motas inalámbricas Nimbus.

Software Defined Radio (SDR)

Un sistema Software Defined Radio (SDR), o radio definida por software, es un sistema de radiocomunicación donde la mayor parte de los componentes se implementan en software en lugar de en hardware. La SDR normalmente está basada en un receptor Zero-IF configurable, de tal manera que puede utilizarse para diseñar distintos componentes (mezcladores, filtros, amplificadores, moduladores/demoduladores, detectores, ...) y sistemas completos (transmisores, receptores, osciloscopios, analizadores de espectros). Los parámetros pueden configurarse dinámicamente, hecho que aporta una gran flexibilidad a la hora de realizar un sistema de radiocomunicación.

Partes de un sistema SDR

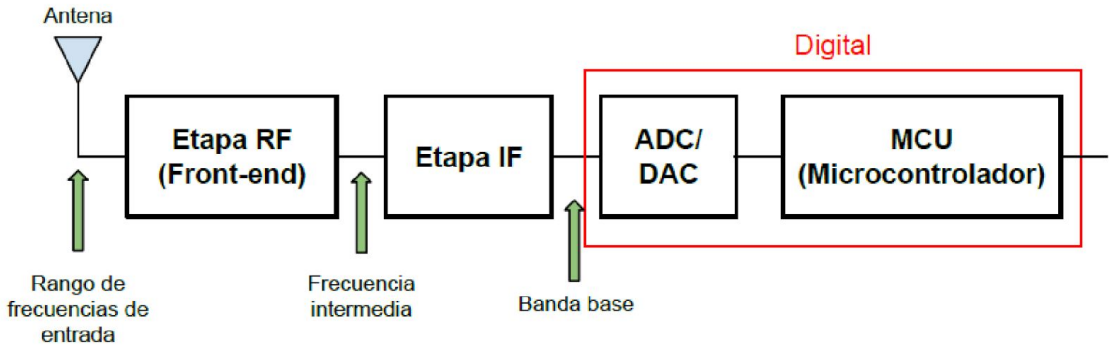


Figura 1. Diagrama de bloques de un sistema SDR

El concepto SDR ha ido evolucionando con los años, pero los SDR se siguen basando en el esquema que se muestra en la Figura 1, y que consta de tres bloques: i) etapa RF, ii) etapa FI y iii) etapa en banda base. La parte de RF e IF se implementan en hardware, mientras que la sección en banda base se hace en software.

La sección de RF, también denominada RF Front-End o cabecera de RF, es la encargada de transmitir/recibir las señales de radiofrecuencia para adecuarlas y convertirlas en frecuencia intermedia en recepción, o amplificar y modular las señales de IF en el caso de transmisión. La frecuencia intermedia puede ser 0, dando lugar al concepto de Zero-IF. Esto es posible gracias a los avances en los componentes hardware. Del mismo modo, la sección de IF se encarga de pasar la señal de IF a banda base y digitalizarla en recepción o pasar la señal de banda base a IF y hacer la conversión digital-analógica de la señal en el caso de la transmisión. Las encargadas de la conversión analógica-digital o digital-analógica de la señal son los módulos ADC/DAC. A su vez, se insertan los módulos DDC/DUC para poder bajar/subir, respectivamente, la tasa de muestreo en el sentido de recepción/transmisión, consiguiendo que la tasa de muestras por la interfaz entre IF y banda base sea inferior. La sección de banda base es la encargada de todo el procesamiento en banda base de la señal. Es donde se lleva a cabo la modulación/demodulación y análisis espectral de la señal. Esta última etapa se lleva a cabo en software.

Conceptos importantes

A continuación repasamos dos elementos muy importantes de un sistema SDR, los conversores analógico-digitales y los digital-analógicos.

- ADCs: Realizan el paso de señales analógicas a digitales asignando a cada nivel de tensión un número digital para ser utilizado por el sistema de procesamiento. Un ADC tiene cuatro características principales:

    - Tasa de muestreo (sample rate): es el número de veces por segundo que el ADC toma una medida de la señal analógica y cuantifica el valor utilizando un número de bits.
    - Rango dinámico (dynamic range): se refiere a la diferencia entre la señal más pequeña (en tensión) y más grande que el ADC puede convertir. Cuantos más bits se utilicen para codificar un rango dinámico determinado, menor será el error de cuantificación.
    - Tiempo de conversión: es el tiempo que necesita el ADC para obtener un número digital a partir de un dato analógico.
    - Número de niveles: indica la precisión con la que se cuantifica un dato analógico y depende del número de bits del ADC.

El teorema de Nyquist determina que, para evitar el efecto de *aliasing* cuando convertimos de analógico a digital, la frecuencia de muestreo del ADC debe ser de al menos dos veces el ancho de banda de la señal de interés. Es decir, para poder reconstruir fielmente la señal analógica a partir de la salida del ADC, salvo por el error de cuantificación, la señal analógica debe ser limitada en banda y la frecuencia de muestreo igual o mayor al doble del ancho de banda de la señal, asumiendo que la señal analógica está en banda base.

- DAC: Un convertidor digital a analógico es un elemento que recibe información de entrada digital, en forma de una palabra de *n* bits y la transforma a señal analógica. Para ello, dada una señal digital

expresada por un conjunto de valores cuantificados en *n* bits asociados a un tiempo de muestreo *Ts*, se puede generar su versión analógica obteniendo una delta de valor el indicado por los *n* bits y filtrando el resultado por un filtro paso bajo ideal, denominado filtro reconstructor. Las características de los DACs son similares a los ADCs, siendo capaces de generar señales de frecuencia máxima igual a la frecuencia de Nyquist. En la práctica, como los filtros reconstructores no son perfectos, la frecuencia máxima que puede generarse adecuadamente es más baja que la frecuencia de Nyquist.

HackRF

La placa HackRF que utilizaremos en las prácticas de laboratorio de la asignatura, es un periférico para radio software que permite la transmisión y recepción de señales en la banda de RF. Puede trabajar en modo standalone, o conectarse por USB a un PC para llevar a cabo aplicaciones concretas. Nosotros la utilizaremos conectada a un PC por USB. Además de esto, algunas de sus características más importantes son las siguientes:

- 10 MHz a 6 GHz de frecuencia de funcionamiento.
- Half-duplex transceiver.
- Hasta 20 millones de muestras por segundo.
- Muestras en cuadratura de 8 bits (8 bits I y Q de 8 bits).
- Compatible con GNU Radio, SDR, y más.
- Software-configurable RX y TX ganancia y filtro de banda base.
- Potencia puerto de antena controlado por software (50 mA a 3,3 V).
- Conector de antena SMA hembra.
- SMA hembra y entrada de reloj para la sincronización de salida.
- Hi-Speed USB 2.0
- Alimentado por USB.

En esta asignatura, utilizaremos la herramienta GNU Radio para crear programas de radio software, que nos permitirán enviar y recibir señales a través de HackRF, modulando y demodulando en software.

La placa ha sido creada por “Great Scott Gadgets” y su autor es Michael Ossman. La versión actual de la placa es HackRF One. La versión anterior se llamaba Jawbreaker. Una de las grandes ventajas de esta placa es que totalmente abierta: disponemos de los esquemáticos, gerbers (archivos para la fabricación del PCB) y todo el firmware de la placa. Toda esta información está disponible en la web, en un repositorio de Github (https://github.com/mossmann/hackrf). Además, en la página web del autor hay un conjunto de tutoriales para iniciarse en el manejo básico de HackRF.

La placa tiene un precio muy competitivo, de 300$ por unidad, lo que ha permitido que se popularice rápidamente. Para hacernos una idea, las placas USRP cuestan al menos más de dos veces que el HackRF.

GNU Radio

GNU Radio es un conjunto de herramientas de desarrollo de software libre (http://gnuradio.org/) y de código abierto que proporciona bloques de procesamiento de señales con los que trabajar en la simulación. GNU Radio está bajo la Licencia Pública General de GNU (GPL) versión 3. Todo el código es copyright de la Free Software Foundation.

GNU Radio es capaz de realizar todo el procesamiento de la señal. Se puede utilizar para escribir aplicaciones, para recibir datos de flujos digitales o para enviar datos en flujos digitales. GNU Radio tiene filtros, códigos de canal, elementos de sincronización, ecualizadores, demoduladores, vocoders, decodificadores, y muchos otros elementos que se encuentran típicamente en los sistemas de radio. Más importante aún, se incluye un método para conectar estos bloques y luego se administra la forma en que se pasan los datos de un bloque a otro. De esta forma, podemos incluso crear nuestro propio bloque y después interconectarlo a otro.

GNU Radio tan solo puede manejar datos digitales. Por lo general, las muestras en banda de base complejas son el tipo de datos de entrada de los receptores y el tipo de datos de salida de los transmisores. El hardware analógico se usa entonces para desplazar la señal a la frecuencia central deseada (modular). Aparte de este requisito, cualquier tipo de datos se puede pasar de un bloque a otro - ya sean bits, bytes, vectores, bursts o tipos de datos más complejos, lo que compone una potente herramienta de procesamiento.

Las aplicaciones de GNU Radio están escritas principalmente en lenguaje de programación Python. Sin embargo, la ruta de datos está escrita en C++.

Para más detalles sobre cómo integrar HackRF y GNU Radio, así como su instalación en un entorno Linux, se recomienda consultar:

- La documentación generada en el Proyecto de Innovación Docente “Receptor Software de bajo coste e Interfaz Computerizada para el Estudio Práctico de las Comunicaciones Radioeléctricas”, presentado en 2014 (http://eprints.ucm.es/27968/)

- La guía de instalación asociada (http://eprints.ucm.es/27968/8/Guia_Instalaci%C3%B3n.pdf).

Cuestiones

• Siguiendo el manual de la práctica 1 propuesta en el anteriormente mencionado Proyecto de Innovación, construya un demodulador AM usando HackRF y GNU Radio.
• Basándose en el proyecto anterior, construya un analizador que le permita observar gráficamente el espectro de frecuencias centrado en 868 MHz.

Motas inalámbricas Nimbus

Además del HackRF, y el soporte software proporcionado por GNU Radio, para llevar a cabo las prácticas, utilizaremos el siguiente material:

• Dos motas inalámbricas Nimbus-868 de la empresa Machine To Cloud Solutions, soldadas sobre una placa de evaluación (Analog v2).
• Programador de ST con placa de expansión de M2C para las motas inalámbricas.
Las placas con las que trabajaremos disponen de una mota radio con un microcontrolador ARM Cortex-M3 de ST Microelectronics, y el tranceiver radio CC1200 de Texas Instruments para trabajar en banda ISM a 868MHz. Los datasheets de ambos componentes se encuentran colgados en el Campus Virtual, y disponibles en los siguientes enlaces:

http://www.ti.com/lit/ds/symlink/cc1200.pdf

http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00078075.pdf

La mota radio está soldada sobre una placa de expansión que dispone de 3 leds, 4 entradas analógicas y 2 pulsadores, que nos permitirán probar algunas funcionalidades básicas de los nodos.

Transceiver radio CC1200

Analice el datasheet con detenimiento y responda a las siguientes preguntas:

• En qué rango de frecuencias puede emitir y recibir el transceiver?
• Si nos centramos en la banda de 868MHz: cuál es la tasa binaria máxima? Podemos configurar cualquier frecuencia de portadora? Con qué resolución?
• Puede el transceiver emitir fuera de la banda asignada ISM Europea?
• Qué modulaciones soporta el transceiver radio?
• Cuál es la sensibilidad del receptor y la potencia máxima del transmisor? Pueden configurarse estos valores?
• Dispone el transceiver de algún mecanismo de encriptación de datos o de transmisión segura? Si dispone de él, descríbalo brevemente.
• Necesita el chip un oscilador externo? Para qué? De qué valor?
• A qué tensión se alimenta el transceiver?

Microcontrolador ARM Cortex-M3

Analice el datasheet con detenimiento y responda a las siguientes preguntas:

• Describa las características del microcontrolador: frecuencia de CPU, tamaño de memoria RAM y de memoria Flash.
• Necesita el chip un oscilador externo? Por qué?
• Cuál es el tamaño máximo del programa en binario que puede ejecutarse?

• De qué interfaces de entrada-salida (comunicación) dispone el chip?
• Cuál es la tensión de funcionamiento del chip?
• Cómo se comunican el microcontrolador y el transceiver?

Placa de desarrollo

La mota inalámbrica va soldada sobre una placa de desarrollo que se alimenta por USB. Responda razonadamente a las siguientes cuestiones:

• Cree que debería existir alguna conversión de tensiones para alimentar el microcontrolador y el transceiver? Por qué? Cómo se lleva a cabo esa conversión?
• Pueden el microcontrolador y la pulga inalámbrica compartir un mismo cristal?
• Cómo se comunicará el microcontrolador de la placa con el PC?

Programación de las motas y entorno de desarrollo

Las motas inalámbricas se programan en C. Para facilitar la programación de las motas, partiremos de un SDK y unas librerías facilitadas por la empresa M2C, así como de unos códigos de ejemplo. Concretamente, estas herramientas nos permitirán:

• Manipular los LEDs y pulsadores de la placa usando funciones en C.
• Comunicarnos con el transceiver radio desde el microcontrolador de forma sencilla (es decir, llamando a una función en C) para enviar paquetes de datos vía radio.
• Utilizar como base un ejemplo de transmisor y receptor sencillo, que solo tendremos que modificar mínimamente.

Los pasos a seguir para tener un entorno de desarrollo son:
1. Instalación del SDK y librerías de la mota Nimbus.
2. Instalación del entorno de compilación (toolchain).
3. Instalación del software para programar las motas (OpenOCD).

Instalación del SDK

En el Campus Virtual encontrará un archivo comprimido con el SDK y las librerías de M2C. El SDK contiene las librerías radio, así como una carpeta con códigos de ejemplo (carpeta “apps”). En particular, encontrará un código de ejemplo de mota receptora y un código de ejemplo para una mota emisora. Más adelante se detalla la funcionalidad del código.

Entorno de desarrollo y herramientas

Las motas son dispositivos empotrados sin sistema operativo (máquina desnuda, aka *bare metal*), por lo que no podemos compilar código directamente sobre el microcontrolador. En estos casos, hay que llevar a cabo una compilación cruzada: el código que se ejecutará en el microcontrolador ARM se compilará en el PC (utilizando la toolchain de ARM). Como resultado

tendremos un archivo binario compilado para ARM (con extensión .axf) que se grabará en la flash del microcontrolador. El micro comenzará la ejecución en el *main* del programa desarrollado.

Instalación de paquetes

Para compilar, es necesario descargar e instalar la toolchain de ARM con soporte para Cortex-M3 (para Linux):

https://launchpad.net/gcc-arm-embedded/+milestone/4.8-2014-q2-update

Para programar el microcontrolador, hay que instalar primero la herramienta OpenOCD versión v0.8.x (para Linux):

http://sourceforge.net/projects/openocd/files/openocd/0.8.0/openocd-0.8.0.tar.gz/download

Para facilitar la instalación de las herramientas, dispone de un script de auto-instalación en el Campus Virtual que instala el SDK, la toolchain de compilación y la herramienta de programación. Descárguese el instalador ("install.tar.gz") del Campus Virtual, así como el SDK y el OpenOCD. Descomprima el instalador en una carpeta en su PC, y desde el terminal ejecute el archivo de instalación mediante: "./install.sh"

Compilación y programación

Los ejemplos proporcionados vienen con sus propios Makefile, de forma que para compilar el programa de una mota inalámbrica, basta con teclear "make" desde el terminal.

Para programar, deberá conectar el programador de ST por USB al ordenador. Y conectar el cable de programación al socket correspondiente en la placa. El microcontrolador a través de la interfaz JTAG. A continuación, desde el terminal teclee "make program" para programar la mota inalámbrica. Una vez acabada la programación del micro, la mota comienza a ejecutar el programa.

Envío y recepción de paquetes radio

Tras instalar todo el entorno, empezaremos por establecer una comunicación radio controlada entre las motas inalámbricas. El objetivo será utilizar una mota como transmisora, con el siguiente comportamiento:

1. Al conectar la alimentación (es decir, conectar el USB) la mota marcada como TX encenderá un LED durante dos segundos.
2. Cuando accionemos el pulsador conectado a una de las entradas analógicas, la mota enviará un paquete de datos.

Por otra parte, conectaremos la mota marcada como RX al puerto USB. Dicha mota tendrá el siguiente comportamiento:

1. La mota receptora estará escuchando en el canal a la espera de recibir un paquete radio.
2. Si recibe un paquete correctamente, envía un ACK a la mota emisora.
3. Cada vez que reciba un paquete, lo enviará por puerto USB al PC, donde se mostrará por pantalla. Para visualizar por pantalla los paquetes enviados, utilizaremos el programa de Linux Cutecom.

Procedimiento

• Empezaremos probando el código de ejemplo de encendido de un LED y envío de un paquete radio de emisor a receptor.
• Compile el código de ejemplo, tanto del transmisor como del receptor, utilizando el comando “make” en la carpeta correspondiente.
• Programe cada binario en la placa correspondiente, utilizando el comando “make program”.
• Alimente la pulga emisora mediante USB a cualquier PC. La pulga emisora debería encender los LEDs un par de segundos.
• Conecte la pulga receptora directamente al USB del PC desde el que verá las trazas. Utilice el comando “dmesg” para ver si se ha reconocido el dispositivo y cuál es el descriptor asociado (generalmente “/dev/ttyUSB0” o “/dev/ttyACM0”)
• Abra Cutecom, y conéctese al puerto “/dev/ttyUSB0” (o “/dev/ttyACM0”), a velocidad de 115200 baudios y sin control de flujo (8N1).
• Compruebe que, si aprieta el pulsador, la mota transmisora envía un paquete que se recibe en la receptora y se visualiza por pantalla.

Cuestiones

• Qué tipo de archivo binario se genera al compilar (puede verlo usando el comando “file”)? Puede ejecutar este archivo en su PC? Por qué?
• Qué datos se muestran sobre los paquetes a través de Cutecom.
• Para facilitar la visualización de paquetes con HackRF, modifique el programa para enviar una ráfaga de 5 paquetes radio en vez de solo 1. Deje algo de tiempo entre envíos (unos 100ms). Encienda el LED verde durante un segundo después de haber enviado los 5 paquetes en el caso de haber recibido correctamente todos los ACKs, y el led rojo en caso de que alguno haya fallado.

Práctica 1: Caracterización del enlace en redes de sensores inalámbricas

En esta práctica se van a estudiar aspectos de la comunicación radio entre los nodos de una red. En primer lugar se va a trabajar con los conceptos básicos de comunicaciones radio. Después se implementará un sistema demodulador y una sencilla red punto a punto. Finalmente se trabajarán los conceptos de potencia transmitida y recibida y se verá cómo estos repercuten en la comunicación de la red. Finalmente, se estudiará el consumo dependiente de la potencia de transmisión y de la ocupación del canal.

Caracterización de la radio:

Por defecto, la radio está configurada para transmitir con la siguiente configuración:

- Frecuencia central 868.1MHz (868.099976)
- Potencia de transmisión 14dBm
- Tasa binaria 50kbps
- Ancho de banda del filtro de recepción 104.17 KHz
- Modulación 2-GFSK
- Desviación de frecuencia 24.94812 kHz
- Conformación del paquete radio: 3 bytes de preámbulo, 16 bits de sincronismo, 2 bytes de CRC.

Cuestiones:

- Partiendo de los diagramas de GNURadio de la práctica anterior, utilice HackRF como analizador de espectros y como osciloscopio para comprobar la frecuencia de recepción de la señal, el ancho de banda y las frecuencias de la señal 2-GFSK. ¿Qué sample rate, frecuencia central y filtro paso bajo ha utilizado en GNU Radio?
- Indique si los valores frecuencias que observa en el FFT Sink son los adecuados.
- ¿Varía la amplitud de la señal si alejamos las motas emisora y receptora? Compruebe cómo varía la señal en el dominio temporal y frecuencial. ¿Cuánto tenemos que alejar las motas para que dejen de recibirse los ACKs? ¿Y para dejar de apreciar la señal en el FFT sink de GNURadio? Viendo estos resultados, ¿qué puede decir con respecto a la sensibilidad de HackRF y de las motas?

Demodulación de la señal GFSK:

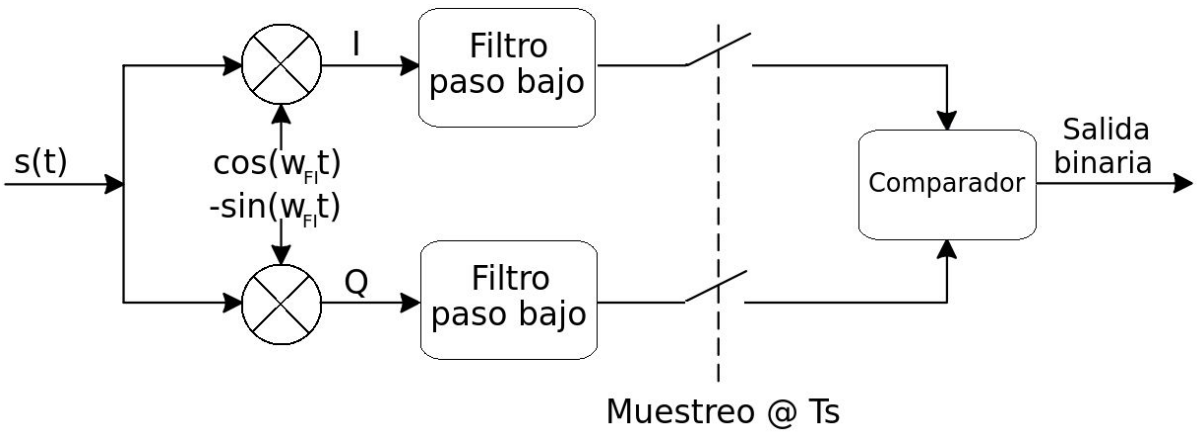
En este apartado se va a demodular una señal GFSK con GNURadio para obtener el mensaje en ella enviado. La modulación GFSK es muy similar a la modulación FSK, pero antes de pasar por el modulador FSK, la señal en banda base (con niveles 1 y -1) pasa por un filtro gausiano que suaviza las transiciones y limita el ancho de banda de la señal. Al final de este documento, en la bibliografía, dispone de algunas referencias con información útil sobre la modulación GFSK.

Se va a utilizar una HackRF a modo de analizador de espectros (que llevará a cabo la demodulación de la señal GFSK) y 3 nodos. Uno de los nodos será, como anteriormente, el receptor; otro será el emisor, y el restante será un repetidor. El repetidor pondrá su sello en el mensaje del emisor. El objetivo final es desvelar la trama ASCII que llega al receptor, pero haciendo uso de la HackRF. El nodo receptor sólamente lo usaremos para el envío de los ACK.

Mantenga la caracterización radio del ejercicio anterior.

Demodulador GFSK mediante HackRF y GNURadio:

Implemente en el SW radio el diagrama de bloques de un demodulador coherente de GFSK, como el que se muestra en la figura.



El filtro paso bajo deberá eliminar la frecuencia de 2*FI que se genera en el mezclador. La señal deberá ser muestreada cada Ts.

Utilice los bloques necesarios para combinar la señal, convertirla a formato binario y guardarla a un archivo de texto que pueda ser leído.

Cuestiones

- Qué ventajas presenta la modulación GFSK con respecto a la modulación FSK? Tiene algún inconveniente?
- Qué bloques de GNURadio ha utilizado para recuperar la información? Cuál es la forma de la señal a la salida de cada bloque?

Nodo repetidor:

Programe el nodo repetidor con el fichero binario "*repetidor*". Asignar identificadores a los nodos y haga que el emisor se comunique con el repetidor. Éste a su vez, cuando reciba un

mensaje lo retransmitirá al receptor. Cuando esto ocurra parpadeará 3 veces un LED del nodo.

Cuestiones:

- Extraiga la trama binaria con GNURadio. Interprete el binario como ASCII y extraiga la información. Como ayuda: sepa que ha de encontrar los códigos ASCII de los identificadores del emisor y el repetidor intercalados en algún mensaje. ¿Cuál es ese mensaje?
- Escriba las ecuaciones de las señales I y Q. Dibuje su espectro.
- Dibuje el espectro de la señal filtrada antes de ser muestreada.
- Dibuje la señal muestreada. ¿Coincide con la salida del comparador?

Reconfiguración de la radio:

Texas Instruments proporciona una aplicación que nos permite ver cómo se generan los paquetes de datos, y cambiar además los parámetros de configuración de emisor y receptor. Puede descargarse el programa Smart-RF Studio directamente la página de TI:

http://www.ti.com/tool/smartrftm-studio

El programa está disponible para Windows de forma nativa y para Mac y Linux utilizando Wine. Una vez instalado, lance el programa, seleccionando el chip CC1200 y en la pestaña roja del chip seleccione “Open RF Device in offline mode”.

Cuestiones:

- Cambie las opciones de conformación de paquete hasta que tenga las mismas características que el paquete que hemos transmitido y guardado.
- A continuación, compruebe que los campos de preámbulo, y palabra de sincronismo coinciden, es decir, que el paquete recibido tiene los mismos bits que el paquete en SmartRF Studio. Compruebe que los campos coinciden y que la longitud del paquete es coherente.
- Qué parte del paquete completo se muestra a través de la interfaz USB-serie de la mota receptora (a través de Cutecom)?

A continuación, cambie las opciones para transmitir:

- Frecuencia central de 868.3MHz
- Con tasa binaria de 10kbps
- Desviación de frecuencia de 25kHz
- Mantenemos modulación 2-GFSK y 14dBm de potencia de transmisión.

SmartRF nos proporciona un “Register View”, que muestra el valor que tendrían que tomar los registros del CC1200 para aplicar esa configuración concreta. Exporte el archivo en el formato “trxEB RF Settings Value Line”.

Los valores generados tienen que sustituirse en el archivo de configuración de cabeceras que está en el SDK: “sdk-nimbus/includes/lib_nimbus_hal/nimbus_rf_link.h”

Ahora deberá recompilar y reprogramar tanto el código del emisor como del receptor.

Cuestiones:

- Compruebe con HackRF que ha cambiado el canal y la tasa binaria. Qué diferencias hay en el espectro de la señal? Cómo ha variado el ancho de banda de la señal?

Caracterización de la potencia de transmisión, RSSI y consumo

En este apartado se pretende que alumno compruebe la relación entre potencia de transmisión y consumo de la etapa radio.
Para la realización de este apartado se necesitará:

- un equipo de medición de corriente o amperímetro
- 1 HackRF
- 3 nodos Nimbus para transmisión y otro más para recepción

Durante los experimentos siguientes vamos comprobar cómo varía el RSSI y el consumo dependiendo de la potencia de transmisión y la longitud del enlace.

Experimento 1: emisor- receptor

Coloque la HackRF como analizador de espectros. Elija la modulación OOK tanto en el emisor como en el receptor. Programe el transmisor para que envíe un paquete (por ejemplo, byte pck = 01010101) cada 5 segundos[1].

Coloque en línea de visión directa el emisor y el receptor. La HackRF ha de estar junto al receptor. Coloque el emisor en una plataforma que pueda desplazar. Coloque el equipo de medición de corriente en el dispositivo transmisor.

Cuestiones:

- Rellene una tabla del Anexo para este experimento. Realice todas las combinaciones con los siguientes juegos de valores:
    - Potencias de transmisión (dBm): -12, -6, 0, 6
    - Distancia (m): 0.5, 1, 2, 4, 8

[1] Nota: las motas reintentan el envío 5 veces (esto se hace en la librería de enlace, sin control por parte del usuario). Cuando el LED rojo se enciende, quiere decir que no se ha recibido ACK tras haber intentado el envío 5 veces. Los 5 envíos pueden verse con al HackRF.

- Dibuje para cada potencia una gráfica de doble eje vertical. El eje de abscisas será la distancia, d. El primer eje de ordenadas será el RSSI y el segundo el consumo. Dibuje los valores medidos.
    - ¿Cómo se comportan las pérdidas, en dB, con la distancia? ¿Puede encontrar alguna relación con la parte de las pérdidas por propagación en espacio libre de la Fórmula de Friis (Eq. 1)? (Tenga en cuenta sólo la diferencia proporcional variando únicamente el parámetro de distancia, d)

$$P_{RX} = P_{TX} G_{TX} G_{RX} \left(\frac{\lambda}{4\pi d}\right)^2 \quad \text{(Eq. 1)}$$

Experimento 2: varios emisores- un receptor

En este experimento se pretende comprobar cómo afectan a una red el que haya más de un dispositivo utilizando el canal.

Vamos a mantener el equipo emisor del experimento anterior con la misma configuración. Este equipo se llamará Nodo N1 a partir de ahora. Pero vamos a añadir un elemento de simulación[2]. Supongamos que este equipo emisor necesita establecer la comunicación a toda costa, y si no consigue el ACK del receptor aumentará la potencia de transmisión "para hacerse oír".
Programe el nodo N1, para que, cuando no se reciba ACK en 5 reenvíos, aumente en +1 dB la potencia de transmisión. Este nodo transmite un paquete cada 15 segundos. El paquete es un contador que empieza en 11000000.

Configuraremos dos equipos transmisores más con la misma modulación OOK y una tasa de transmisión de 1 paquete cada 100 ms para uno (N2) y 250 ms para otro (N3). Los mensajes que enviarán, respectivamente, serán 00000010 y 00000011. (Si se anima, puede colocar más de dos nodos.)

Coloquemos los emisores formando, por ejemplo, un triángulo isósceles; todos con visión directa con el receptor. El N1, el más alejado del receptor, a una distancia de 4 m.

Cuestiones:

- Rellene una tabla del Anexo para este experimento. Las medidas de consumo se refieren al nodo N1. Deje los otros nodos transmisores apagados. Para una distancia de 4 metros del receptor y con los siguientes juegos de valores, mida el consumo y el RSSI:
    - Potencia, P_N1, de transmisión (dBm) del nodo N1: -6 < P_N1 < +6 con incrementos de +1 dBm

[2] ¿Sabía que, en un entorno masificado como en el metro, cuando muchos individuos intentan acceder a descarga de contenidos, el ancho de banda disponible por equipo móvil (léase *smartphone*) se reduce? Esto hace que la descarga de cualquier contenido sea más lenta y por ende se necesite más tiempo que si la estación base estuviese dedicada solamente a nosotros. Esto se traduce en un aumento del consumo de la batería.

- Rellene una tabla del Anexo para este experimento. Las medidas de consumo se refieren al nodo N1. Encienda los otros nodos transmisor. Para una distancia de 4 metros del receptor del nodo N1, realice todas las combinaciones con los siguientes juegos de valores y mida el consumo y el RSSI:
    - Potencia mínima de transmisión (dBm) del nodo N1: -12
    - Potencia máxima de transmisión (dBm) del nodo N1: +12
    - Potencia de los nodos restantes (dBm): 0

- A la vista de las tablas que ha rellenado:
    - ¿Qué conclusiones puede sacar al respecto de esta red?
    - ¿Cuándo se ha llegado al contador más alto en recepción? Si existiese un contrato entre el emisor y el receptor con una QoS, Ud., como equipo emisor, ¿qué solicitaría para mejorar su QoS?
    - Elimine del escenario el nodo N2. ¿Tarda más o menos en aumentar el contador en recepción? Si fuese el equipo receptor que garantiza la QoS, ¿qué le contestaría a la solicitud anterior del emisor?

Bibliografía:

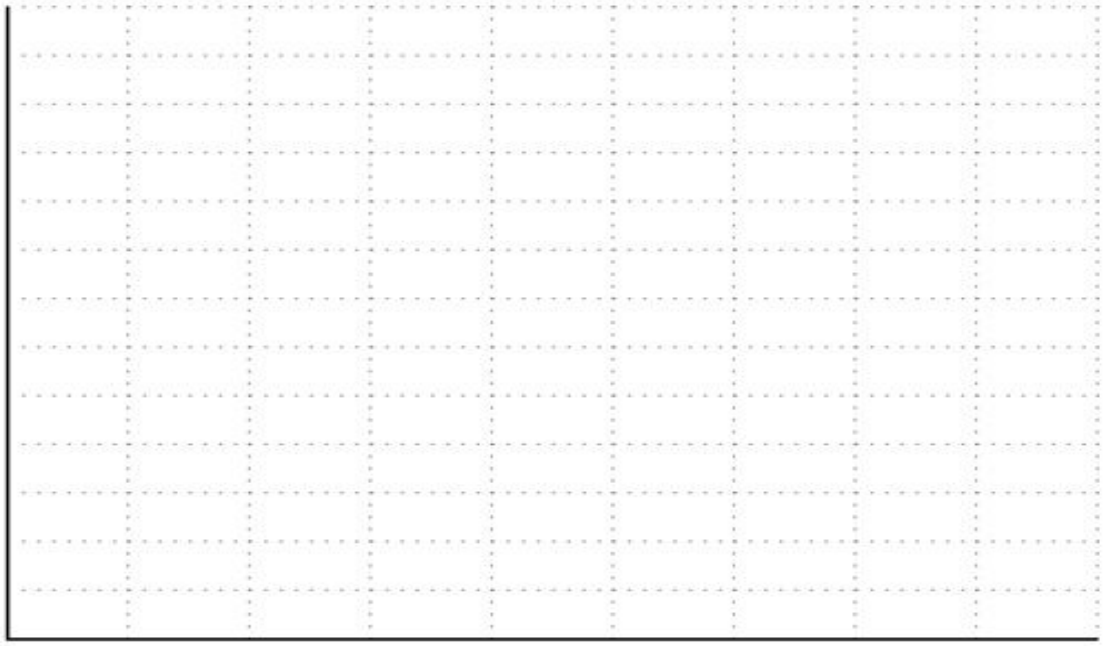
[1] K. Munir, O. Olsen, “Design of an Integrated GFSK Demodulator for a Bluetooth receiver”, Capítulo 4, http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/5479/pdf/imm5479.pdf

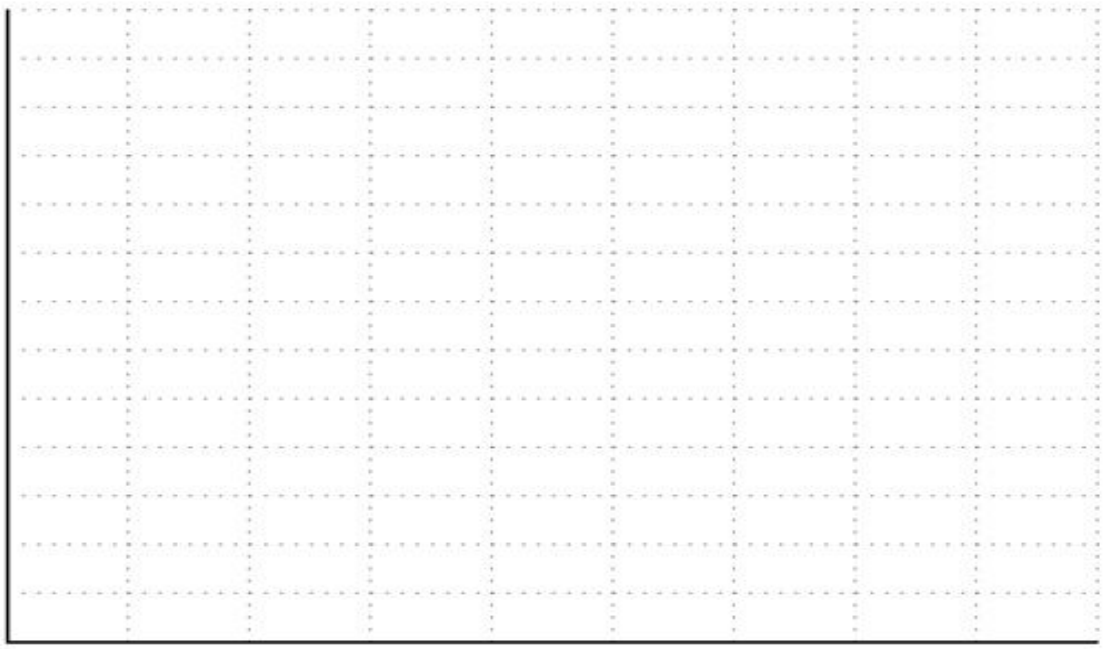
[2] S.H. Gerez, “Implementation of Digital Signal Processing: Some Background on GFSK Modulation” http://wwwhome.ewi.utwente.nl/~gerezsh/sendfile/sendfile.php/gfsk-intro.pdf?sendfile=gfsk-intro.pdf

Anexo. Tablas.

Experimento:			
Potencia TX (dBm)	**Distancia (m)**	**RSSI (dBm)**	**Consumo (mA)**

Gráficas





Práctica 2: Análisis de topologías de red

En esta práctica se van a estudiar aspectos básicos sobre topologías de red entre nodos. Para ello, se probarán dos topologías de red (topología en estrella y en anillo), y se analizarán las ventajas e inconvenientes de los protocolos de comunicación síncronos y asíncronos.

Envío de paquetes en topología de estrella.

Decimos que esta es una de las topologías más comunes para redes de sensores, en las que todas las motas emiten sus mensajes hacia un concentrador (que generalmente actúa como una pasarela de datos).

Inclusión de 5 motas transmitiendo hacia un receptor. Cada mota transmite un paquete con el identificador de la mota. Hay que fijar manualmente la dirección de origen y de destino de cada una.

La mota receptora deberá ser capaz de recibir paquetes del resto de motas e imprimir el paquete recibido por el interfaz serie, para visualizarlo a través de cutecom.

Que se encienda el LED rojo si no hay ACK de nivel de enlace.
Nota: las motas reintentan el envío 5 veces (esto se hace en la librería de enlace, sin control por parte del usuario). Cuando el LED rojo se enciende, quiere decir que no se ha recibido ACK tras haber intentado el envío 5 veces. Los 5 envíos pueden verse con al HackRF.

Cuestiones:

- Aumentar la tasa de envío de paquetes y comprobar con la HackRF si aumenta sensiblemente la ocupación del canal.
- Comprobar también si aumenta el número de retransmisiones a través de Cutecom.
- Analice el consumo de la red. Qué nodo es el que domina el consumo? Por qué?

Cambio de topología de red: Topología de anillo asíncrono.

Cambiamos la topología de la red para que las motas se envíen un paquete en anillo (fijamos las direcciones manualmente). Utilizamos 6 motas, identificadas con números del 0 al 5.

A cada salto, incrementamos en 1 el byte del payload que coincide con el identificador de la mota, de forma que cada mota del anillo cambie únicamente un byte del payload. De esta forma podremos comprobar fácilmente que el paquete ha pasado por todas las motas. La sexta es la encargada de imprimir el paquete por Cutecom (es la que tiene un firmware distinto).

El envío empieza en la Nimbus0 y el paquete debe llegar a la mota Nimbus5 correctamente.

Algo así como:

- Nimbus0: envía el paquete “000001”
- Nimbus1: espera a recibir un paquete de la Nimbus0 y lo modifica “000011”
- Nimbus2: espera a recibir un paquete de la Nimbus1 y lo modifica “000111”
- ...

Por tanto, la Nimbus5 deberá recibir, en la primera vuelta del anillo, un “111111”. Una vez recibido el paquete, la Nimbus5 se lo reenviará a la Nimbus0, que incrementará el byte correspondiente “111112”. Al final de la segunda vuelta, la Nimbus5 recibirá un “222222”. Dado que trabajamos con bytes, al llegar a 255, el contador deberá reiniciarse.

Para poder observar mejor el comportamiento, las motas esperarán 2 segundos entre envíos.

Cuestiones:

- Pueden las motas dormir entre envíos de paquetes? O deben estar constantemente escuchando en el canal? Por qué?
- Cómo afecta esto al consumo de las motas para estos chips en concreto? Cómo varía el consumo de las motas con respecto a la topología de estrella?

Topología de anillo síncrono.

Con el objetivo de mejorar la eficiencia energética de la topología de anillo asíncrono, a continuación implementaremos una topología síncrona. El objetivo de esta topología es conseguir que las motas inalámbricas puedan dormir durante parte del tiempo, en vez de estar constantemente escuchando el canal a la espera de mensajes.

Para ello, todas las motas conocerán el tiempo de envío entre mensajes, que estableceremos de nuevo en 2 segundos. El funcionamiento y procesamiento llevado a cabo por cada mota es el mismo que en el caso anterior. La diferencia radica en que cada mota, tras recibir un mensaje, se dormirá durante 1.5 segundos y luego escuchará en el

Cuestiones:

- En base a los consumos teóricos en transmisión, recepción y modo dormido de las motas, calcule cómo mejora el consumo energético global de la red al implementar la topología síncrona en comparación con la asíncrona. Y si se enviara un paquete 1 vez por minuto en vez de cada 2 segundos?
- Por qué hemos establecido un tiempo dormido de 1.5 segundos en vez de 2 segundos? Qué problemas de sincronización existirían en la red si durmieramos 2 segundos? Podemos seguir teniendo estos problemas aún manteniendo este tiempo de guarda de 0.5 segundos?
- A qué componente hardware se deben dichos problemas? Si el tiempo de envío entre paquetes consecutivos aumenta, se soluciona el problema?